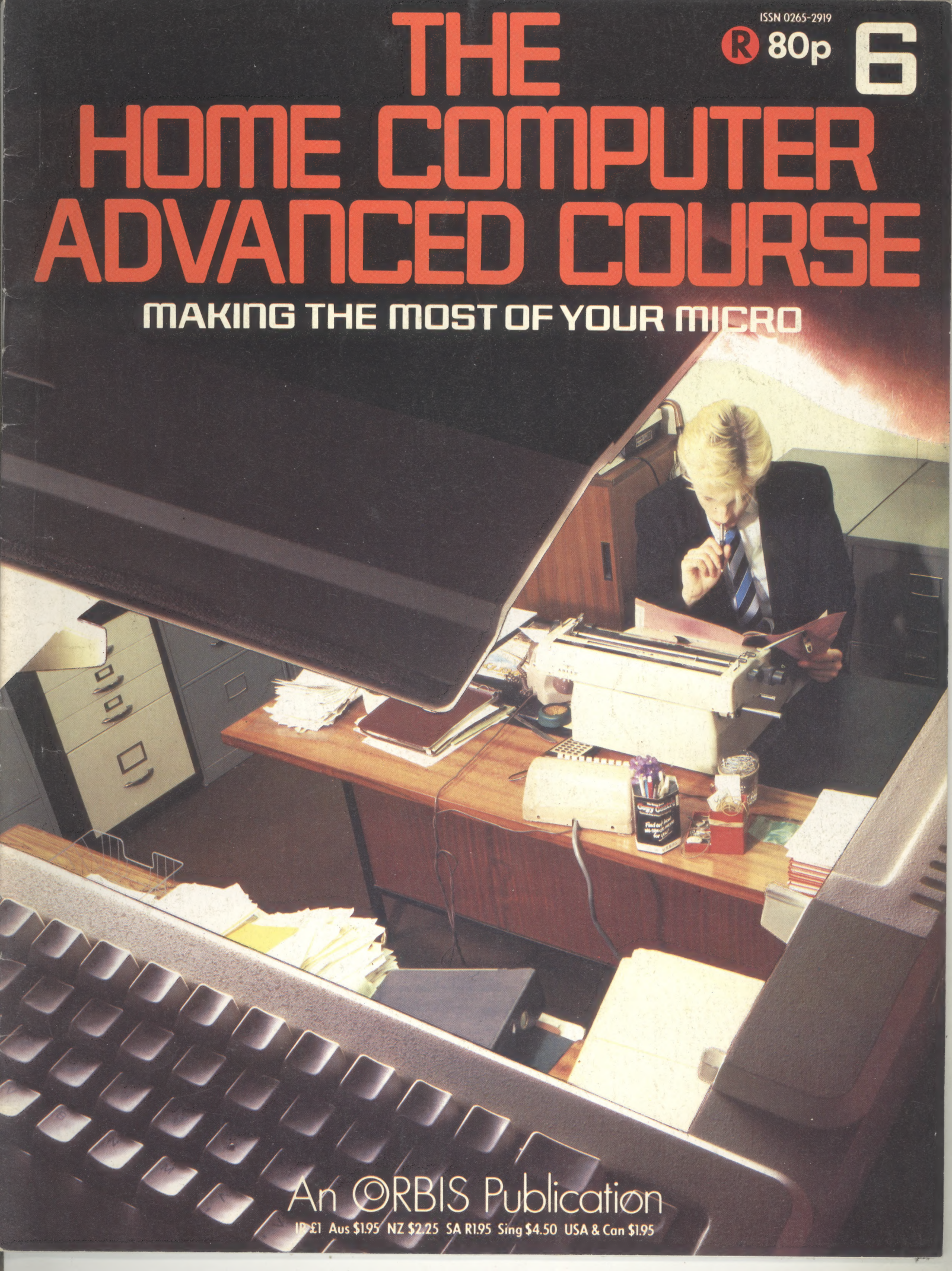
ISSN 0265-2919

R 80p

6

# THE HOME COMPUTER ADVANCED COURSE

## MAKING THE MOST OF YOUR MICRO



An ORBIS Publication

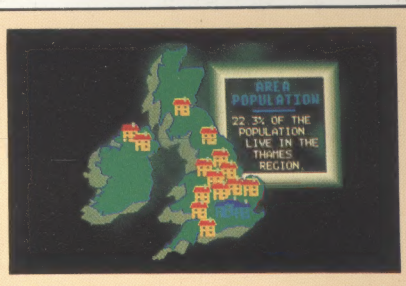
IR£1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

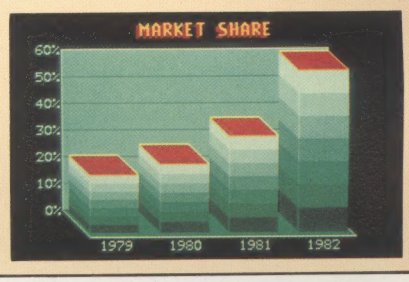
# CONTENTS



## Next Week

- It has been said that inside every Dragon 64 there lurks a Dragon 32. We take the lid off the 64 and see just what makes it tick.
- In the next instalment of Workshop we'll continue our look at electrical components and show you how to calculate the value of resistors.
- For a relatively small outlay, home computer owners can now generate their own viewdata graphics – for pleasure or profit.







**Editor** Jonathan Hilton; **Art Director** David Whelan; **Deputy Editor** Roger Ford; **Production Editor** Catherine Cardwell; **Staff Writer** Brian Morris; **Picture Editor** Claudia Zeff; **Designers** Hazel Bennington, Julian Dorr; **Sub Editors** Robert Pickering, Keith Parish; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Researcher** Helena Siedlecka; **Contributors** Lisa Kelly, Steven Colwill, Richard King, Martin Hayman, Tony Harrington; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Co-ordinator** Ian Paton; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

**HOME COMPUTER ADVANCED COURSE** – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

**How to obtain your copies of HOME COMPUTER ADVANCED COURSE** – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

**Back Numbers UK and Eire** – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

**How to obtain binders for HOME COMPUTER ADVANCED COURSE** – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

**Note** – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

COVER PHOTOGRAPH BY MARCUS WILSON-SMITH

# EASY ACCESS


IAN McKINNELL

**In the early 1980s British Telecom — operators of Prestel, the largest public viewdata system in the country — combined with the computer manufacturer and publisher Prism Technology to use the system as a medium for distributing software to home computer users and as an exchange point for 'electronic mail'.**

Micronet 800 — so-called because its original site was page 800 of the Prestel database — allows owners of a variety of home computers direct access to a considerable range of software that can be run and, perhaps more important, stored on tape or disk for later use, often free of charge. Subscribers, however, need ancillary hardware and software.

Because one is accessing a remote database via the public switched telephone network, the first requirement is for a modem (modulator/demodulator), a hardware device that translates the computer's signals into signals capable of telephonic transmission. There are two options — hard-wired modems and acoustic couplers. The essential difference between them is that a hard-wired modem is connected into the telephone system by means of a plug, whereas the acoustic coupler operates with a telephone handset (British Telecom standard issue), turning the signals into

**Tuning To Micronet**
A BBC Micro can be connected to Micronet using any Prestel compatible system. The easiest way to do this is to buy a complete package of hardware (a modem or acoustic coupler), software (a ROM, cassette or disk) and Micronet/Prestel subscription from Prism Microproducts. On this BBC, the Micronet ROM is positioned so that the computer is ready for Micronet the moment you switch on

audible tones, and vice versa.

Any modem compatible with Prestel may be used to access Micronet 800 — that is, it must transmit at 1,200 baud and receive at 75 baud. Such modems are available from a variety of sources, but Micronet recommends units supplied by Prism. These are of three main types. Modems 1000 and 2000, which are suitable for most of the home computers for which software is available from Micronet, offer two-way transmission (1,200 baud transmit and receive — useful for communication between microcomputer users as well as to Prestel), and the standard 75/1,200 baud rate. In the case of Modem 2000 the changeover is controlled by software. The Acoustic Modem, also available for a range of machines, is restricted to 75/1,200 baud operation. Its advantage is that it does not require a special connection point and may be used on any telephone — even a public one if the pre-payment is sufficiently high.

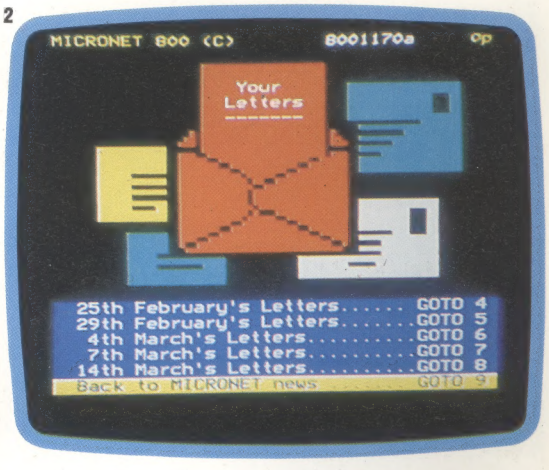
The third type of modem available from Prism is custom-built for the Sinclair Spectum and is known as the VTX 5000. Like Modems 1000 and 2000, it offers both full-duplex (transmissions in both directions simultaneously) 75/1,200 baud and half-duplex (one direction at a time) 1,200/1,200 baud operation, but it also includes all the

**1 Full Features Index**
The full features menu gives a good overview of the information available and lets you choose which page to look at next

**2 Letters**
One of the most popular areas is a users' letters page — both the letters and Micronet's replies are often a little cryptic for newcomers

1

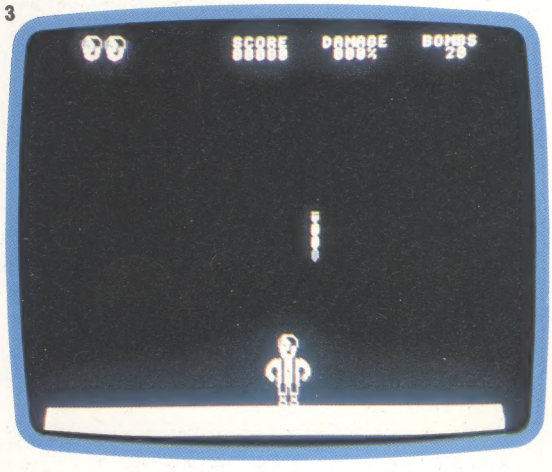

2



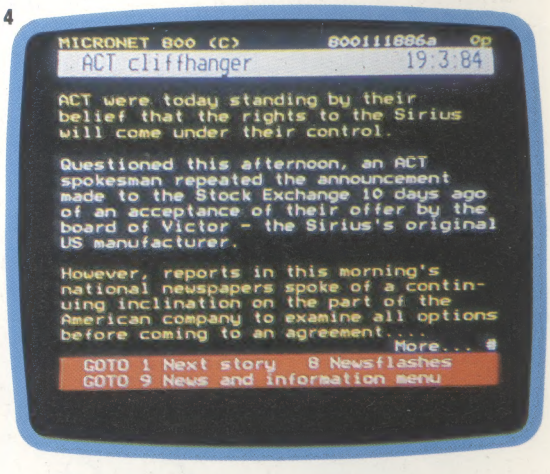

**3 Free Games**
You can read programs off Micronet into your own computer — this game was free, others have to be paid for

**4 News**
Micronet has news. It can update the pages every day so it's often ahead of weekly magazines

3




4




**5 Charles and Diana**
The Gallery is a series of Teletext masterpieces — there are also games, jokes and so on for your entertainment

**6 Weather**
Subscribing to Micronet gives you access to lots of other Prestel pages with information such as the weather page

5


6


LIZ HEANEY

necessary communications software in ROM form — a much less expensive method than those offered to users of other types of microcomputer, who must buy operating software as well as the ancillary hardware.

This software comes in two forms: on disk, or — for the BBC Microcomputer — as a replaceable ROM. BBC users can opt to have the system self-booting by installing the ROM in the right-most socket. On other computers the functions are the same once the program has been loaded from disk. As the systems are so similar in operation, we will refer in general to Micronet as implemented on the BBC Microcomputer.

The operating/de-coding software, created by Scicon, one of Britain's longest-established software houses, is necessary to allow individual microcomputers to operate according to Prestel protocols, which Micronet uses throughout (the Micronet subscriber automatically becomes eligible to access the entire Prestel database, with its wealth of information on all manner of subjects).

## LOGGING ON

All viewdata systems are completely menu-driven. The user is presented with a series of choices, each of which represents an exit path from the frame being accessed. As the user enters the Micronet operating system the first series of frames asks for identification codes. These are in two parts, consisting of a 10-digit personal identifier and a Prestel subscriber number. Since Prestel allows the user to order goods and services, the price of which may then be charged to the user's account, these numbers function in their own right as a credit card. In the case of Micronet, the numbers determine who is charged the cost of downloaded software. A good deal of thought has obviously gone into the generation of this security system, and it would appear to be foolproof. For instance, the user can conceal his code number from onlookers by entering 10 asterisks in its place — whereupon the system demands that it be entered again, but this time does not display it.

Once logged on to the system, the user is free to wander around the database, collecting and sending messages, downloading games or business software at will — and also uploading, for Micronet is a two-way system with scope for selling software as well as buying it.

And so into the Micronet database itself. To be accessible from a series of menus, the structure of a database must be hierarchical — a 'trunk and branches' arrangement — and Micronet is no exception. Once the user is successfully connected to the Micronet computer via the telephone system, the first choice presented defines the general area in which he will work — 10 subject headings such as What's New, Computermart, Talking Back or Mailbox/Telex. While the titles are more or less self-explanatory, there are two 'exits': one that leads to a Help page, and another that leads into Prestel itself.

One of the main features of Micronet is its ability to distribute software. How is that operation performed? The appropriate menu prompt is Telesoftware, and taking that route presents the user with a list of the types of computer for which software is available (each has its own exit route), together with a menu that offers 'Top Ten' charts of the most popular items, an explanation of how to sell software through Micronet, Help lines and an advertisement.

Following the BBC Microcomputer route, we are led to a frame that offers a choice between individual software packages (games, business packages, utilities), new releases, bestsellers, free programs and Help frames. Selecting one of the four options offering programs reveals a list of titles with short descriptions. Choosing one of these prompts Micronet to display the frame known as the Downloader Menu. This defines the options available, which are to download and run the program, log-off the system, or run and stay connected to Micronet. Staying connected can be expensive in terms of telephone charges if you do anything other than save the newly acquired program on tape or disk.

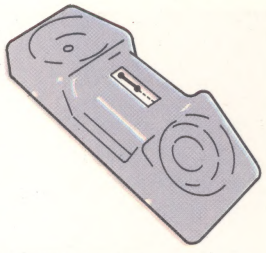
While it is true that Micronet offers free software, the amount is limited, although Micronet claims to have available more than a hundred programs at any one time. Many of the free programs are amateur productions, so high quality sound and graphics should not be expected every time. The other software available is charged at varying rates, the price being set by the author.

As one has to pay for much of the Micronet software, this aspect of Micronet should perhaps be viewed as an alternative software distribution system rather than as a public service. Indeed, Prism's efforts to find a more effective method of distribution do not end with Micronet — the company is the UK agent for the American Romox system, which programs ROM cartridges for sale at retail outlets.

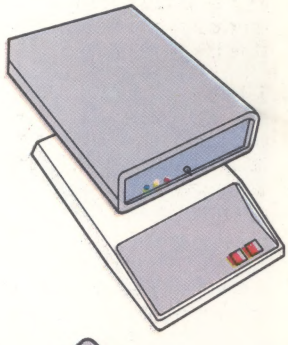
The other functions of Micronet are similar to the Prestel system itself, of which it is part: Mailbox is replicated in the larger system, as are the news and advertisement pages.

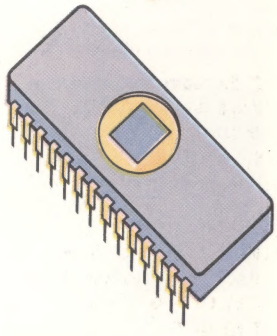
The route through the hierarchy, as determined by the exit chosen by the user, is defined by a frame identification number. It is possible to bypass the sometimes long-winded sequential access route by calling up any frame directly, if its identification number is known. This is a useful attribute for the experienced user, who is likely to become frustrated at having to turn through page after page of unwanted detail.

Considering the quantity and quality of the services it offers, Micronet is not expensive. A subscription costs £1 per week for private users at home and slightly less than £2 for business users and schools. There is the additional cost of the modem and, of course, the local telephone call that connects the user to the Micronet or Prestel computer — plus the cost of any piece of software one buys.

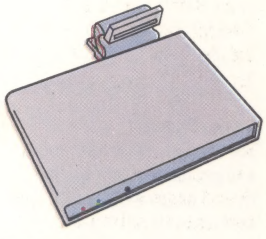


**Acoustic Couplers**
An acoustic coupler provides the simplest way to connect machines such as the BBC, RML 380Z and the Apple to Micronet. You'll need appropriate software for your machine but the list of machines supported is continually growing





**Modems**
Modems 1000 and 2000 plug directly into the phone system and so provide a more reliable way to access Micronet. Appropriate software for the BBC can be supplied on a ROM chip so it's available the moment you switch on



TONY DUNCAN-SMITH

**For The Spectrum**
The VTX 5000 is a modem specially designed for the Spectrum and has all the software you need built-in. It sits under the Spectrum and connects to its expansion connector

# WHEELS OF FIRE

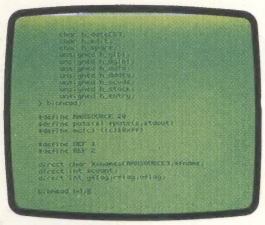
**With the Dragon DOS, which Dragon Data supplies with its disk drives, the company has maintained its policy of providing easy-to-use commands, couched in Microsoft Advanced Color BASIC. Up to four drives can be connected to the Dragon 32 or 64 via the parallel interface cartridge supplied, which also contains the DOS in ROM form.**

Dragon drives use 5¼in, single-density disks. These are magnetically formed, on command, with 40 tracks, divided into 18 sectors, each containing storage space of 256 bytes. This gives a total storage capacity per disk of 40 × 18 × 256 = 184,320 bytes or 180 Kbytes. The DOS requires part of the storage space for file identification, disk management and the directory leaving about 175 Kbytes of formatted space.

The directory is a list of all files contained on a disk. On command it displays the file names, file types, the number of bytes in the file and the number of bytes free. Four types of file are possible: BASIC program, data, binary and backup. The type of file stored is indicated by appending .BAS, .DAT, .BIN and .BAK, respectively, to the relevant file names. The Dragon DOS does not mention how the files on a disk are distributed in memory and accessed, but as there is no command to close up free space on a disk there is probably a form of BAM (Block Availability Map).

Strangely, the Dragon DOS has no means of accessing data randomly, a byte at a time. This is a useful facility for a program that needs to read small amounts of data frequently — a database, for instance. Instead, the DOS uses what it calls 'simulated random access', which appears to do the same thing except that a few extra lines of BASIC code are required to implement it. The way Dragon DOS is implemented complements BASIC extremely well.

A relatively high purchase price and small storage capacity make this disk system only average value for money. Perhaps conscious of this, Dragon Data plans to increase its range of drives to include 80-track double-sided, double-density drives at reasonable cost. For the Dragon 64 an optional RAM-resident operating system, booted from disk, is being introduced. Called OS9, this takes a hefty 16 Kbytes of RAM but provides an advanced 6809 operating system that includes an exceptionally powerful DOS and has PASCAL, C, COBOL and 09 structured BASIC available on separate disks.

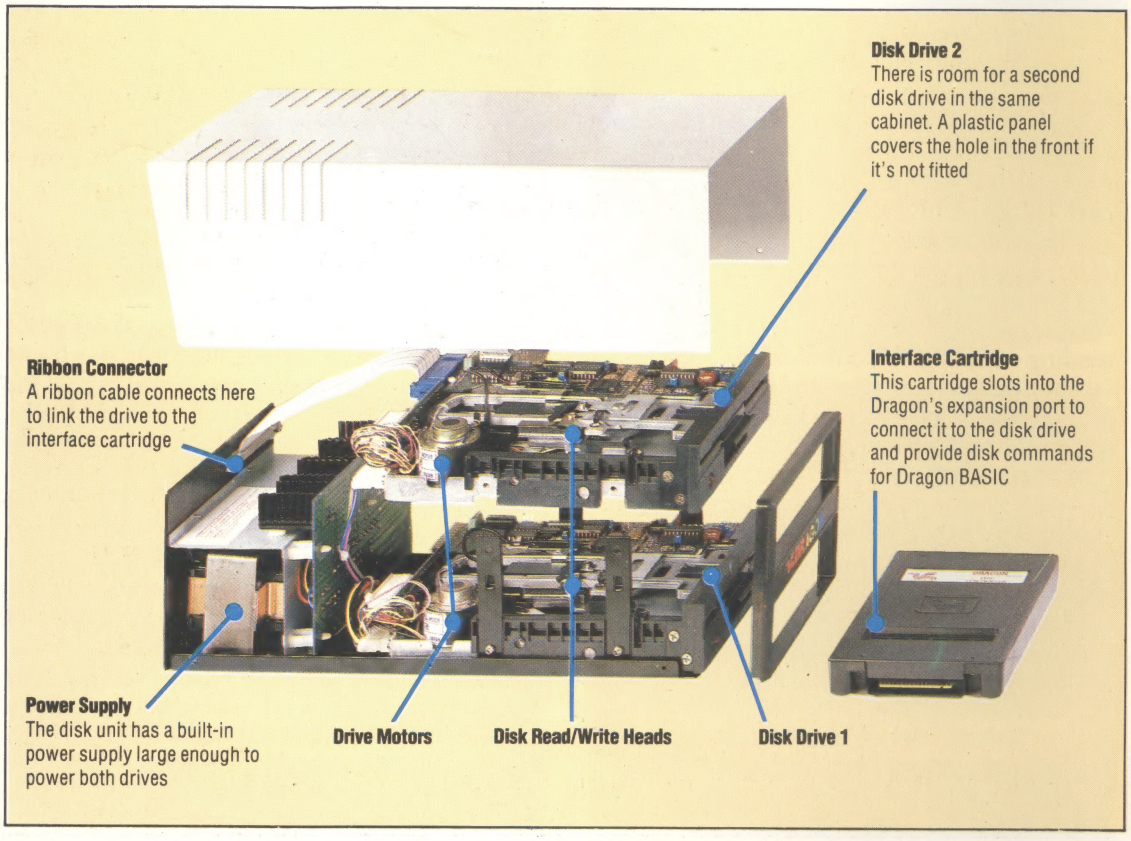


IAN McKINNELL

**Language Options**
Dragon's first batch of disk software runs under the OS9 operating system and includes advanced computer languages such as PASCAL and the C compiler shown here

**Sophisticated Package**
OS9 also enables you to run sophisticated business software. This word processing package includes a spelling checker with a 42,000 word dictionary and the ability to produce 'mailshot'-style letters

**Driving Power**
The Dragon disk drives come in a single cabinet with its own built-in power supply and room for two drives. A ribbon cable connects this to an interface cartridge which then plugs into the Dragon. The disks work with Dragon's own operating system or the more expensive and professional OS9 software range. Each disk has room for 175K of data




IAN McKINNELL

# Dragon Disk Commands

When writing commands the parameters describing the file are often the same. In this case the most frequently used command format is:

COMMAND"D:FILENAME.TYP"

where D selects a drive (1-4) and overrides the default drive; FILENAME is up to eight characters that specify the file; and .TYP is the file identifying code, which may be specified by the user, but if not then the DOS will assume the default value: .BAS. This format is represented by FSP.

**DRIVE N**
In this case N can be 1 to 4. This selects the default drive.

**DSKINIT**
This command formats the specified disk with:

DSKINIT D,S,T

in this case D selects the drive; S selects the side of the disk (either 1 or 2) to be formatted, default being 1; T selects the number of tracks to format (40 or 80), default being 40. With a standard single-drive system it is necessary to type only DSKINIT (ENTER) and the disk inserted will be wiped and formatted correctly.

**DIR D**
This command displays the directory of the specified drive, D, as follows:

```
DIR
PROGNAME .BAS 1654
M/C BINARY .BIN 1389
PROGDATA .DAT 2581
PROGNAME .BAK 1654
167322 FREE BYTES
```

## PROGRAM FILES

**SAVE**
This will store program or binary files to disk. SAVE FSP stores a program file. It is unnecessary to specify .BAS as this is the default value. The command:

SAVE FSP,SSSS,EEEE,XXXX

stores a binary file. In this case, SSSS is the start address in decimal of the code to be stored; EEEE is the end address, and XXXX is the address the program is executed from.

**LOAD**
This will read program or binary files from disk with the command:

LOAD FSP

If the file specified is a binary file, FSP can be supplemented by ,SSSS. This will indicate the new start location in memory of the binary file.

**RUN FSP**
This will immediately load and run the specified BASIC program.

**CHAIN FSP**
This will load and run a program without changing previously stored variables. This is especially useful for programs that share data. Adding ,SSSS will have the same effect as for the LOAD command previously described.

**FREE D**
This displays the number of bytes free on a specified drive

**COPY**
This will duplicate the file OLDFSB as NEWFSB, as follows:

COPY OLDFSB TO NEWFSB

If drive numbers are unspecified, the copy is made on the same disk on the default drive.

**RENAME**
This changes the name, but not the type, of a file. Both FSPs must refer to the same drive or default.

**MERGE FSP**
This command will superimpose a specified BASIC program file over another held in memory, with the effect that the programs are merged. The program previously held on disk will take precedence if line numbers are duplicated.

**KILL FSP**
This will delete a specified file from disk.

**PROTECT**
This command 'software-protects' a file from erasure or overwriting by any command, except DSKINIT, with:

PROTECT ON FSP

This will also cause a reverse field P to be displayed with the file name on the disk directory. The command PROTECT OFF FSP will remove protection.

**BACKUP**
This will copy the entire contents of a disk in one drive (DA), onto a disk in another (DB) with the command:

BACKUP DA TO DB,S,T

S and T take values exactly as for the command DSKINIT, enabling copies to be made on different format disks and drives. The user of single disks can simply type BACKUP (ENTER), whereupon instructions are displayed for alternating source and destination disks.

**VERIFY**
ON and OFF control this command, which automatically checks that the contents of a stored file are the same as the original.

## DATA FILES

**FWRITE**
This is used to create and write a data file containing variable lists. For each FWRITE, a channel is opened to the specified file, allowing further data to be appended to the data already stored. It is written as:

FWRITE"FILENAME";VAR

In this case FILENAME is the file to be written, or created and written to, and VAR is a variable list containing the data to be stored. Commas and colons terminate variable strings in data files, unless they are intended to be read by FLREAD as strings. The command:

FWRITE"FILENAME",FROM SB,FOR TB;VAR

writes VAR to FILENAME starting at byte SB, extending the length of the list to a total of TB bytes. Only 10 files accessed by FWRITE can be open at one time.

**CLOSE D**
This closes channels to files opened by FWRITE, FREAD and FLREAD on a specified drive.

**CREATE"FILENAME",FL**
Creates a data file, FILENAME, of FL bytes in length.

**FREAD**
This command is constructed as FWRITE. VAR is read into memory or, taking the second example given, VAR is read starting from byte SB. The read pointer is then advanced by SB+TB bytes.

**FLREAD**
This is constructed in the same way as FREAD, except that commas and colons are not read as terminators.

**EOF**
This is used to indicate the final valid entry in a file being read. For example:

EP=EOF("FILENAME")

where EP is zero, until the read pointer reads the final record and changes EP to one.

**LOC"FILENAME"**
This will display the position of the read pointer as the number of the next byte to be read from a specified file.

**SWRITE**
This command will store data in S sector on T track as the two strings A$ and B$ to a maximum of 128 bytes each, as follows:

SWRITE D,T,S,A$,B$

**SREAD**
This will retrieve data stored by SREAD using the same format. A$ and B$ can be given different variable names.

# CIRCUITOUS ROUTES

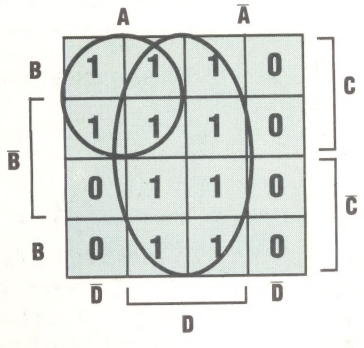
**Having investigated the use of Karnaugh maps in the simplification of Boolean algebra expressions using two and three variables (see page 92), we now look at the more complex cases involving four variables. We also give examples of how k-maps are used to simplify the process of designing electronic circuitry.**

*Four Variables:* In the cases of Boolean expressions involving four variables, the k-maps—and the expressions themselves — can seem formidably difficult. But by applying the simple ideas established when we looked at two and three variable k-maps, they soon become extremely familiar and easy to manipulate.

For example, suppose we are asked to simplify this expression:

$$ABC\overline{D} + ABCD + \overline{A}BCD + A\overline{B}C\overline{D} + \overline{B}CD + A\overline{B}CD + \overline{A}\overline{B}CD + B\overline{C}D$$

We can see that a four variable k-map is required, but, although there are eight components of this expression, we will actually need to fill in 10 ones in the k-map (both the $\overline{B}CD$ and $B\overline{C}D$ terms represent two cases each). Thus the k-map is:




From the k-map we can see that the central group of eight ones represent every possible combination involving D. The group of four ones in the top left corner includes all the possible cases involving A AND C. So, the expression simplifies to A AND C OR D (expressed as: A.C + D).

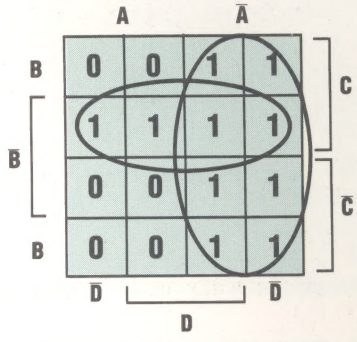
Sometimes an expression may need some initial manipulation to arrange it into a form suitable for representation on a k-map, as in the following example:

$$\overline{A + B + C} + \overline{A}.B + \overline{B + C}$$

Here we must first apply de Morgan's law (see page 46) to the expression before it is possible to draw up a k-map. It is reformulated as:

$$\overline{A}.\overline{B}.\overline{C} + \overline{A}.B + \overline{B}.C$$

and the k-map this expression produces is:




From the k-map this expression is seen to simplify to: $\overline{A} + \overline{B}.C$. Using de Morgan's law again, the expression is finally simplified to:

$$\overline{A.(B + \overline{C})}$$

## CIRCUIT DESIGN

### Example 1: Thirty Days

You probably know the rhyme: 'Thirty days hath September, April, June and November...'. Let us suppose that each month of the year is encoded into a four bit binary code—0001 for January through to 1100 for December. Our task is to design a circuit that will accept the four bit code as an input, and output a 1 if the month input has thirty days.

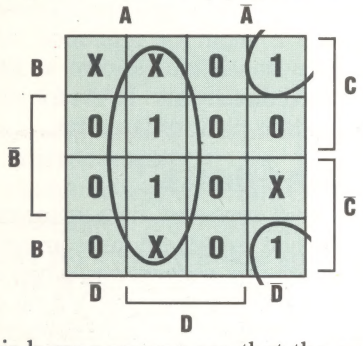
The truth table for such a circuit is:

| MONTH | INPUTS | | | | OUTPUT |
|---|---|---|---|---|---|
| | A | B | C | D | S |
| | 0 | 0 | 0 | 0 | X |
| JAN | 0 | 0 | 0 | 1 | 0 |
| FEB | 0 | 0 | 1 | 0 | 0 |
| MAR | 0 | 0 | 1 | 1 | 0 |
| APR | 0 | 1 | 0 | 0 | 1 |
| MAY | 0 | 1 | 0 | 1 | 0 |
| JUN | 0 | 1 | 1 | 0 | 1 |
| JUL | 0 | 1 | 1 | 1 | 0 |
| AUG | 1 | 0 | 0 | 0 | 0 |
| SEP | 1 | 0 | 0 | 1 | 1 |
| OCT | 1 | 0 | 1 | 0 | 0 |
| NOV | 1 | 0 | 1 | 1 | 1 |
| DEC | 1 | 1 | 0 | 0 | 0 |
| | 1 | 1 | 0 | 1 | X |
| | 1 | 1 | 1 | 0 | X |
| | 1 | 1 | 1 | 1 | X |

The output X in the truth table signifies an invalid input. We shall assume that the circuit will *not* receive such signals. From the truth table, wherever S = 1 we can form the following Boolean expression from the binary bits of the input:

$S = \bar{A}.B.\bar{C}.\bar{D} + \bar{A}.B.C.\bar{D} + A.\bar{B}.\bar{C}.D + A.\bar{B}.C.D$

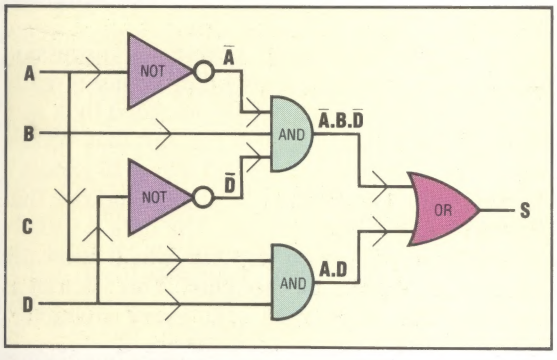
Drawing these on a k-map, together with the 'invalid input' conditions (X), gives us:




From this k-map we can see that the expression reduces to:

$A.D + \bar{A}.B.\bar{D}$

And thus, our 'thirty day month' signal circuit can be constructed:




**Example 2: Odd Numbers**

Given that the numbers 0 to 15 can be coded by four binary digits (0000 to 1111), we are asked to design a circuit that will accept the four bit code as an input and output a 1 if the output represents an odd number greater than two.

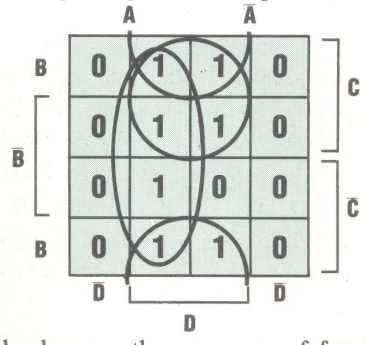
The first thing we must do is set up a truth table for all the possible conditions:

| DECIMAL NUMBER | INPUTS A | B | C | D | OUTPUT S |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 1 | 1 |
| 8 | 1 | 0 | 0 | 0 | 0 |
| 9 | 1 | 0 | 0 | 1 | 1 |
| 10 | 1 | 0 | 1 | 0 | 0 |
| 11 | 1 | 0 | 1 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 | 0 |
| 13 | 1 | 1 | 0 | 1 | 1 |
| 14 | 1 | 1 | 1 | 0 | 0 |
| 15 | 1 | 1 | 1 | 1 | 1 |

From this truth table we form the following Boolean algebra expression, for all the conditions where S is true (= 1):

$S = \bar{A}.\bar{B}.C.D + \bar{A}.B.\bar{C}.D + \bar{A}.B.C.D + A.\bar{B}.\bar{C}.D + A.B.\bar{C}.D + A.B.C.D$

The Karnaugh map for this expression is:



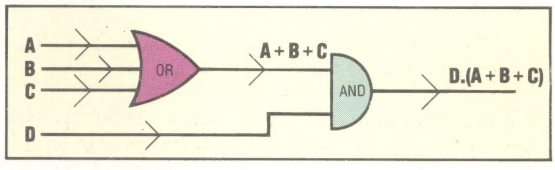

From the k-map, three groups of fours can be isolated, represented by this expression:

$S = A.D + C.D + B.D$

and this can be further simplified using the distributive law to get:

$S = D.(A + B + C)$

Consequently, the circuit can be designed:




In the next instalment, we will review the more important aspects of the Logic course so far, and provide a comprehensive set of review exercises.

**Exercise 5**

**1)** Simplify the following Boolean expressions, using Karnaugh maps:

**a)** $A.B.C + A.\bar{B}.\bar{C} + \bar{A}.\bar{C} + \bar{A}.B.C + A.B.\bar{C}$

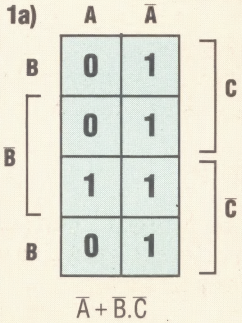
**b)** $\overline{B + C} + B.\bar{C} + A.C$

**c)** $A.\bar{B}.D + \bar{A}.D + A.B.C.D + A.B.\bar{C} + \bar{A}.B.\bar{C}.\bar{D}$

**2)** A circuit is to be designed that will accept the binary representations of the whole numbers between 0 and 7 inclusive. The circuit is to give an output if the number input is odd or if it is a multiple of three (i.e. 3 or 6). By drawing a truth table and obtaining a simplified expression, draw a logic circuit that will carry out this function.

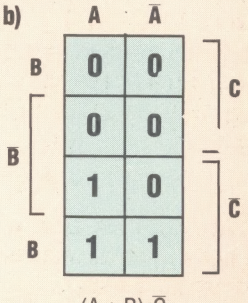
**Answers to Exercise 4 on Page 93**

**1a)**




$\bar{A} + B.\bar{C}$

**b)**




$(A + B).\bar{C}$

# A

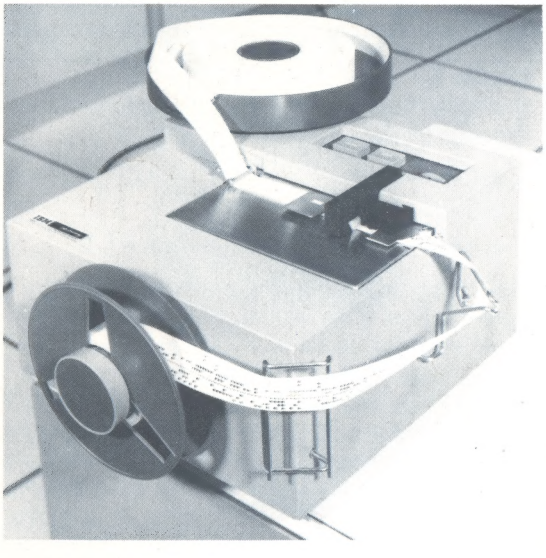
## ASCII

The American Standard Code for the Interchange of Information is the system adopted by most microcomputer and software manufacturers for representing alphanumeric characters in binary form. There is no fundamental reason why the letter A should be represented by 01000001 (as it is in this system), since the CPU is not aware of whether it is processing letters, numbers or symbols. The main benefit of agreeing on a standard code is that programs can then be written to run on several different machines. Furthermore, data can then be sent from one computer to another, either by means of a telephone link or stored on a magnetic medium.

ASCII is the most popular system, but by no means the only one. Another is EBCDIC – the Extended Binary Coded Decimal Interchange Code – which was favoured by IBM on its early mainframe machines, though not the IBM PC.

Many standard commercial packages for microcomputers have the facility to produce *ASCII files,* the functions of which are best illustrated by example. A document stored as a file from a word processing package will usually contain a number of special control characters — to indicate that a heading should be centred on a page, for example. If an ASCII file is generated, all these control characters will be removed, and replaced with pure ASCII characters, such as the correct number of spaces. An ASCII file can then usually be read in to another word processing package that perhaps uses a different set of control characters.

**ASCII Code**
The best place to see a coding system like ASCII in action is on an old-fashioned paper tape machine, where each character is represented by a row of holes across the tape



COURTESY OF IBM

## ASSEMBLY LANGUAGE

Assembly language, in essence, is a more readable way of expressing machine code, using alphanumeric labels instead of hexadecimal addresses and mnemonics instead of numbers for the opcodes. However, unlike other higher-level languages, Assembly language (sometimes confusingly referred to as 'assembler') translates byte-for-byte into machine code. Consequently, there is no loss in efficiency through writing in Assembly language first. Once the program is complete, it is passed to the assembler program, which translates the Assembly language labels and mnemonics into their machine code equivalents. The Assembly language program (known as the source code) is filed for reference, while the machine code translation (known as the object code) is loaded directly into RAM for immediate execution and/or filed for later use.

## ASYNCHRONOUS

There are two types of data transmission – synchronous and asynchronous – and the terms are most commonly encountered when referring to serial data transmission over a long distance, usually by means of a telephone line. *Asynchronous transmission* is the simpler to implement, and is the basis for interfaces such as the RS232, which are found in many home computers. Synchronous transmission, however, is the more efficient in terms of maximum data transfer rate, and is used on all large computer systems.

In an asynchronous system, transmission commences whenever the sending device is ready to issue the data. Each byte is preceded by a *start bit* — just an additional bit at the start that 'wakes' up the receiving computer and tells it to expect a further eight bits of real data, at the baud rate that has been set on both devices. The data is usually followed by one or two *stop bits,* which allow the receiver to 'collect its thoughts'. This pattern is followed for every byte (i.e. every character) transmitted.

On a synchronous system, both devices feature very accurate clocks, which are synchronised, and transmissions occur only at specified intervals. It's a bit like saying to someone: 'I will be waiting by the telephone at the start of every hour for five minutes in case you call' — although in the computer the interval will be in terms of microseconds. The result is that the start and stop bits are no longer needed and efficiency is thus increased. The main difficulty with synchronous transmission is the need to keep both clocks in perfect synchronisation.

## ATTRIBUTE

The screens on early microcomputers could only display text in upper and lower case (along with a few graphics symbols if you were lucky). Now some quite elementary home computers allow the programmer to specify each character's colour, the background colour on which it is printed, whether it is normal or extra bright in intensity, and whether steady or flashing. These specifications are called the *attributes* of a character, and they are usually stored in a single byte of RAM for each screen location. Each bit, or group of bits within that byte, will look after one of those attributes, and they can either be determined using the BASIC commands such as BRIGHT, FLASH, INK, PAPER and so on, or by manipulating the bits directly using the AND and OR functions of machine code.

# BARE BOARDS

**Not all microcomputers are housed in carefully designed eye-catching cabinets. A variety of machines, ranging from the simple to the quite sophisticated, are sold as bare boards. One of the most versatile of these is Rockwell's Advanced Interactive Microcomputer, better known as the AIM 65, which is intended as a teaching and development aid.**

At its simplest, the AIM 65 is a bare board, devoid of casing. However, from the operational point of view, it is very comprehensive and is the only development machine that has a printer built in. Most development systems are much less generous in their design. They don't have a proper keyboard, nor do they have the 20-character, 16-segment LED display that gives the AIM user a useful window into the machine.

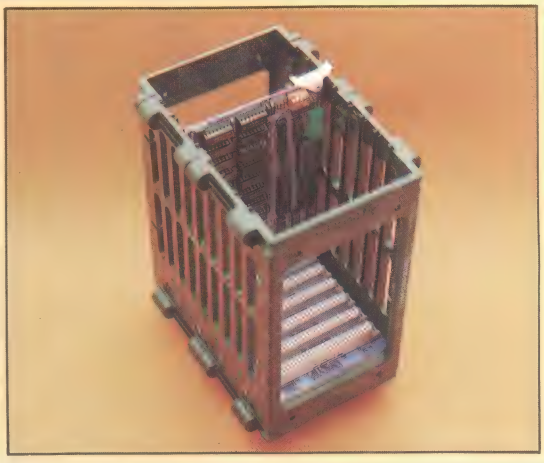
These three features are not essential, but the more usual sort of machine — with a hexadecimal keypad, eight-character, seven-segment display and no printer — is far more difficult to operate. The essential requirements are input/output channels and memory, and in these areas the AIM is more typical with a standard four Kbytes of RAM and 12 Kbytes of ROM; however, in this respect it is better equipped than some other development machines, which often have only one Kbyte of RAM and two, four or occasionally eight Kbytes of ROM.

Compared with business and home computers, almost all of which have a minimum of 16 Kbytes of RAM and frequently much more, a mere four Kbytes may seem meagre. However, the AIM 65 also has two good solid expansion connectors that allow you to add more boards, such as the 32 Kbyte static RAM board. In general, development systems are used in applications that don't require a large memory capacity, and it's surprising how much you can do in the three Kbytes or so available after the operational requirements of the machine have been satisfied.

Since the AIM 65 is built around the 6502 microprocessor, all the parts of the memory map have fixed boundaries. This demands that the bottom 512 bytes are RAM to accommodate the machine stack, which occupies byte locations $0100 to $01FF. Zero page, from $0000 to $00FF, is a special area used by the 6502 as 'pseudo-registers'. Because the address decoders don't have


CHRIS STEVENS

**Take AIM . . .**
A complete AIM system can be a little untidy, although for neatness the keyboard and motherboard can be supplied in a case. This is ideal if you are using the AIM in its basic 4K form. For expansion, one extra card can be plugged straight into the back of the AIM. If you need more, you have to use an expansion chassis and connect it to the computer using the two cards and ribbon cable shown in the foreground. The blue metal box is the power supply



**Expanded Memory Board**
You can go on expanding the AIM almost forever — this is an expansion chassis to hold a further eight circuit boards (there are also four and 16 slot sizes). The boards are a standard size and connect with the standard 96-pin Euro connector. You can buy cards for memory expansion (up to 128K of RAM and ROM), an IEEE standard interface and cards to allow the AIM to be connected to floppy disk drives, monitors and full size printers

to include the page address (binary 00000000) in the instruction, the latter become, in effect, shorter and thus much faster to execute than regular ones. This feature is part of the secret of the 6502's power and popularity, as it gives the chip 256 user-definable registers.

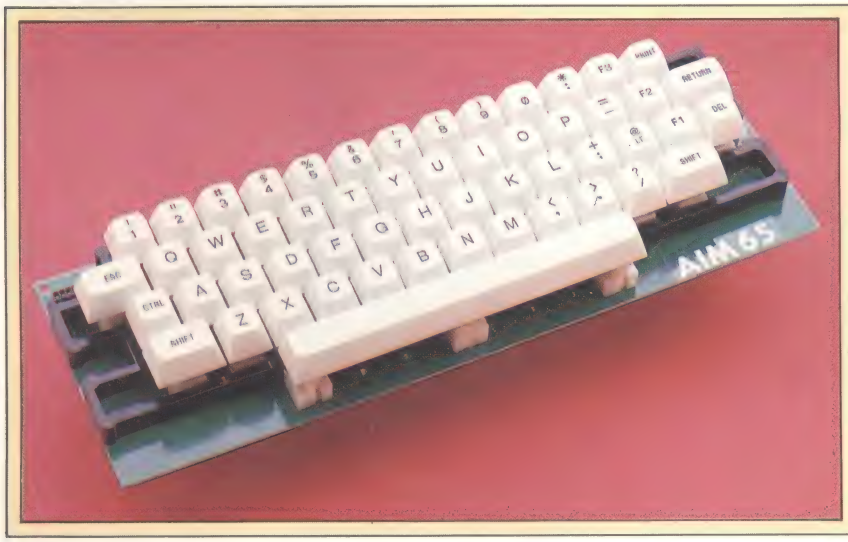
As noted, the AIM has quite a lot of ROM compared with other development machines. This is placed at the top end of the memory, again because that is where the 6502 locates it. A selection of languages and utility programs are available, though all of them depend on the monitor ROM, a four-Kbyte block of low-level system utilities that resides at $E000 to $EFFF. It contains a line editor, single stepper (which executes the program one instruction at a time so that the memory contents can be inspected and, possibly, changed at every stage) and tracer (which displays the contents of the program location register at every step during execution), as well as the usual register and memory alteration features.

The machine comes supplied with five ROM sockets, into which a variety of firmware options can be plugged. Among these are a version of BASIC, which has all the main functions, providing five-byte floating-point numbers and single-line defined arithmetic functions.

Much more relevant to the potential buyer of the AIM 65, who is likely to be interested in control and monitoring of industrial processes, or other dedicated applications, are the other ROM sets. These include an assembler and an interesting but little-known language called PL/65, which looks a bit like ALGOL or PL/1 and compiles into assembler source code (see page 116). This can then be manipulated for fine-tuning and assembled by the assembler.

INSTANT PASCAL, an unusual language that has been strangely neglected, is available to the AIM 65. Unlike almost all other PASCAL dialects it is interactive and interpreted, providing almost all the benefits of structuring as well as the convenience and flexibility of BASIC. However, PASCAL is an extensive language and won't fit into the AIM 65 unless it is expanded. FORTH is also available, giving much the same advantages. As this demands rather less of the machine than PASCAL, it can be run without expansion.

Since development machines are intended to be used by engineers, often as controllers or monitoring devices for complex machinery, their hardware and I/O features are more critical than




**Keyboard**
The AIM has an efficient keyboard that allows the standard machine to be easily programmed in BASIC and other languages — many equivalent systems have only hex keypads (just 0-9, A-F and a few other keys) and so aren't so easily usable. PRINT in the top right-hand corner is used in conjunction with the CTRL key to turn the built-in printer on and off. With the printer switched on, everything that appears on the display is also printed on the printer

CHRIS STEVENS

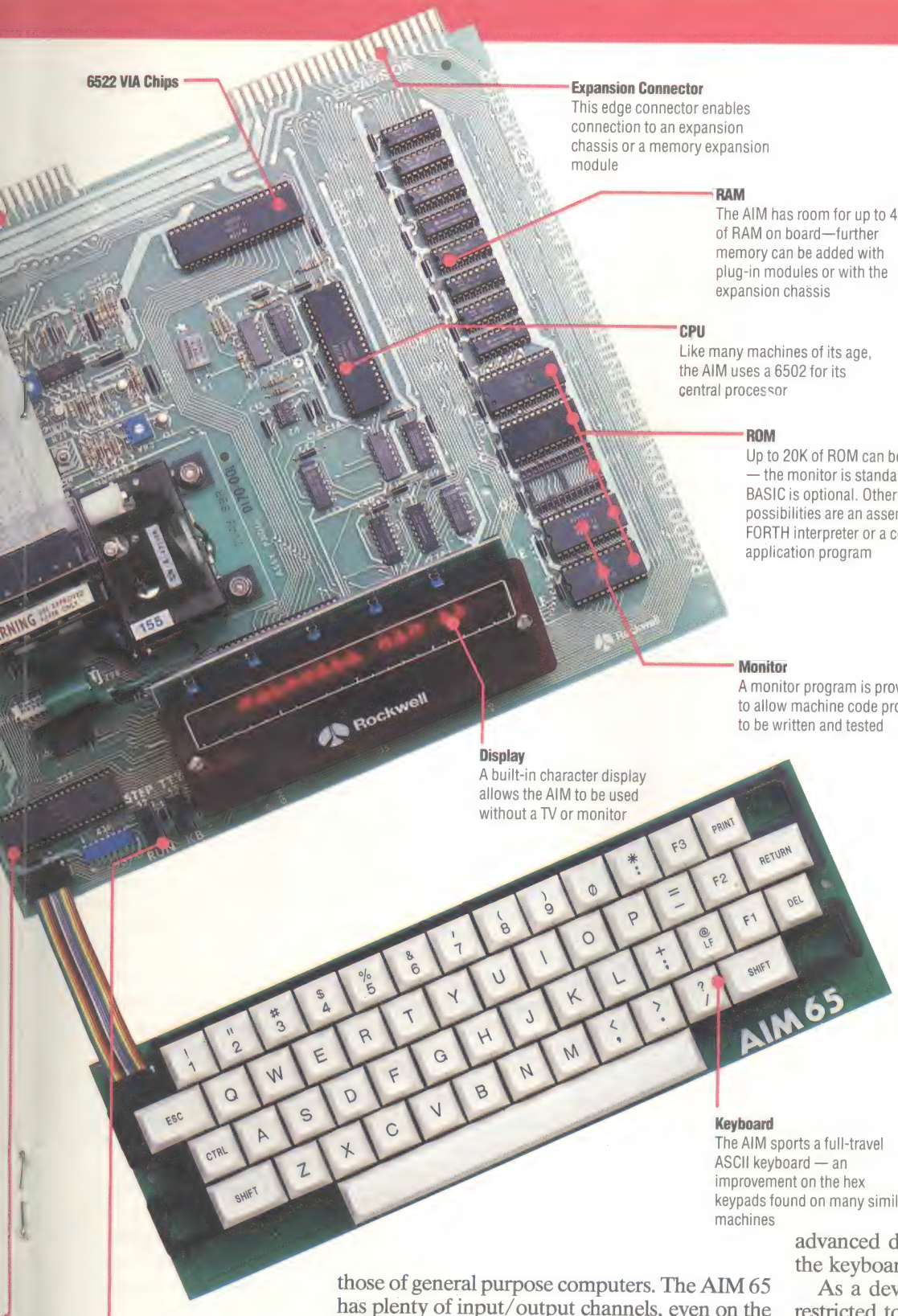
**Application Connector**
This edge connector has two ports for connecting the AIM to whatever devices it is to control

**Power Connector**
The power supply connects here; the AIM needs the usual 5v for the computer and 24v to drive the printer

**Printer**
The AIM is very unusual in providing a built-in 20 column thermal printer — useful for control applications that require a permanent record of events

**Reset Switch**
Used to restart the computer

**Keyboard Decoder**
A 6532 RIOT chip controls the keyboard




## AIM 65

**PRICE**
£404.28 plus VAT

**DIMENSIONS**
292×267×60mm

**CPU**
6502

**MEMORY**
4K ROM, with sockets for up to 20K, and 4K RAM, expandable to 64K

**SCREEN**
No video display, but a 20-character, 16-segment LED display. A controller card is needed to establish a normal VDU output

**INTERFACES**
Two eight-bit bi-directional ports, each with two control lines, plus system bus

**LANGUAGES AVAILABLE**
A mini assembler and line editor are supplied, and a full assembler, BASIC, FORTH, PL/65 and INSTANT PASCAL are available

**KEYBOARD**
53 typewriter-style keys, including three function keys

**DOCUMENTATION**
Installation and hardware manuals that are a model of clarity and completeness, with every detail the user could require

**STRENGTHS**
Its most important advantage is the extreme flexibility of the machine, attributable to its expansion capabilities and access to a variety of languages. The documentation is superlative

**WEAKNESSES**
It has no serious weaknesses. There is a relative lack of high-level software, but then the AIM 65 is a developer's and designer's machine rather than an applications machine

those of general purpose computers. The AIM 65 has plenty of input/output channels, even on the basic unit.

Apart from the expansion connector, which carries all the main signals, including the data and address lines, clock signals and power lines, there are two 6522 Versatile Interface Adapter chips, one of which is used to control the printer, teletype interface and cassette interface. The other is uncommitted and appears on the J2 application connector as two eight-plus-two-line bi-directional ports. A 6532 RAM I/O Timer (RIOT) is also provided, but this complex and advanced device is dedicated solely to handling the keyboard.

As a development machine the AIM 65 isn't restricted to running in BASIC or any other high-level language. It is anticipated that users will program their own specialised ROMs, which can then be put into the ROM sockets, thus dedicating the machine to a specialised job.

Overall, the AIM 65 is a rugged, flexible, well-supported and well-designed single-board computer. It has sufficient facilities to make it attractive to anyone requiring a small but very useful machine without the features of a standard business computer, but with more flexibility than is usually found in a home computer.

# OFFICE MANAGERS

**The availability of cheap computing power, coupled with the high cost of developing applications from scratch, has created a market for 'off-the-shelf' business packages. In this series we will be looking at how these systems work. The market spans both domestic and small business users, companies and corporations.**

The label 'business packages' covers a wide range of applications, and in addition to differences of function there are crucial differences of scale. The first main division of scale is between cassette-based software aimed at the home computer user and disk-based software. Because disk storage technology gives programmers access to fast read and write facilities, all business software ought to be disk-based, but not everybody wants to pay an additional £200 to £400 for a floppy disk-based system.

The other division is that between single-user microcomputers and multi-user machines — microcomputers that are capable of supporting a number of separate terminals. We will start with the simpler systems and deal with the multi-user applications later in the services.

Business software is designed to be used by the layman rather than by the computer specialist. Because of this, whether a program is designed for a cassette-based system or the largest multi-user disk system, it will usually take an 'interactive' approach. This means that the user is guided through all stages of operating the system by a series of prompts and messages on the screen.

Typically, the various operations will be arranged as a numbered set of options on a 'menu'. Selecting a function from the main menu might take the user to a secondary or even a tertiary menu of choices. The screen will then display a request for data, usually information in a form similar to what would be entered into a manual book-keeping system. This 'user-friendly' approach has been one of the factors that have made business applications programs so successful.

The three functional divisions that we will consider in this series are the sales ledger, the purchase ledger and the general (or nominal) ledger. The sales ledger records sales made by the business in the ordinary course of its trading. The purchase ledger records the purchases necessary to enable it to continue trading. The general ledger provides a total picture of the state of affairs of the company and its bank balance at any given time.

Business programs are composed of files. Each file contains a number of records and each record contains a number of fields. The distinction between files, records and fields is simple. A sales ledger program, for example, will have a customer master file consisting of the records of each of the customers. Details on each customer, such as the name and address, are divided into fields within the record.

In addition, of course, a business program needs a way of performing operations on data. Examples of programming routines needed are input routines (which allow data to be entered into the system) and arithmetical routines, which manipulate numerical values in numerical fields inside records and across records.

In 'mainstream' business packages — those designed for use with disk-based systems — the sales and purchase ledgers record a great deal of detail. Sales and purchases are recorded (the accounting term is 'posted') against individual customer and supplier accounts. The idea is that the package should store as complete a record as possible of all the transactions between the


**Money Management**
A sensible approach to the limitations of cassettes is to sell separate programs that can be used together if necessary. This invoicing program works quite happily by itself, leaving you to manually update your own stock records. Alternatively, it can use files generated by its companion programs as part of a completely automated system. One advantage of this approach is that you can gradually computerise your business step-by-step rather than undergoing a complete change all at once


**Easiledger**
Cassette-based business software is limited as to the size and number of programs in a package. As a result, programs like this one tend to concentrate on a particular task. This package has elements of accounting software such as debit/credit ledgers, year-to-date summaries and so on. But it's only intended as a management aid for controlling cash flow and won't handle your whole accounts by itself

LIZ HEANEY

business and each of its customers and suppliers.

Even the biggest microcomputer-based system, however, has an upper limit on the amount of data that it can store, and cassette-based packages cannot hope to deal with the details of as many transactions as disk-based packages. So one of the key questions any potential user of cassette-based software (or disk-based software, for that matter) has to ask is: Can this system cope with the volume of work associated with my business?

The more data that is stored in a file, the longer it takes a computer to sort, so the usual approach is for programs to retain detailed information on outstanding transactions only (an outstanding transaction is one where money is still owing). Systems based on this principle are called 'open item' ledgers.

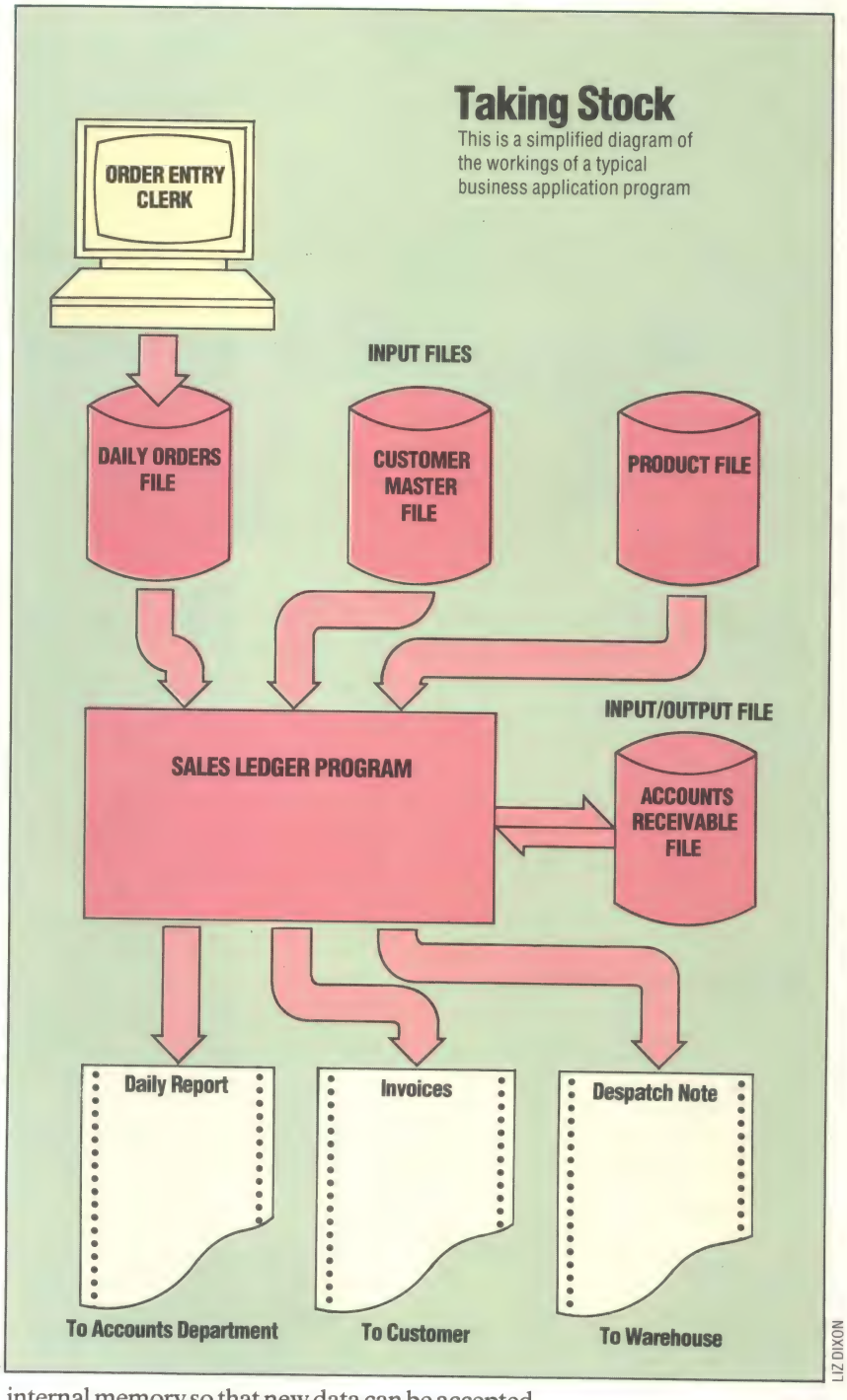
An alternative approach is for the computer to retain only the cumulative value of transactions between the business and its customers or suppliers. This is called 'balance brought forward' accounting. It conveys less information, but requires less memory and is simpler to operate. In other words, there is a trade-off between efficiency (in terms of the level of detail that can be recorded) and the demands made on the machine's processing and memory capabilities.

As we will see in the course of this series, disk-based sales and purchase ledger programs have a number of specialised features that it is not possible to incorporate in cassette-based packages. Sales ledger programs, for example, might include an invoicing option that allows the business to generate invoices and statements to send to its customers. Purchase ledger programs might have the ability to print cheques as well as remittance advices (listing what has been purchased) to send to suppliers.

Business software carries out a variety of functions. The easiest way to understand them is to consider the way cash flows into and out of a business. The basic requirement of anyone in business is to find out how much income the company brings in and how much it costs to operate the business. One of the simplest systems for achieving this is the cash book. A manual (non-computerised) cash book has its pages divided into as many columns as the business needs to identify the way in which it spends or receives its cash. The cash book will also have to keep track of the VAT value on each receipt and each payment so that it can pay over or reclaim the correct amount of VAT from Customs & Excise.

The cash book might be updated on a daily or a weekly basis. Each transaction might be written up, or, more usually, the total value of a day's takings might be entered. The difference between the cash book and a full ledger-based accounting system consisting of sales, purchase and nominal ledgers, hinges on the cash book's lack of detail.

Cassette-based business/book-keeping programs have a limited amount of working memory at their disposal. They cannot write data to disk and then clear an area of work space in their internal memory so that new data can be accepted. Because of this, cassette-based programs tend to follow the summary format of the cash book.




Instead of having three separate cassette programs for the sales, purchase and nominal ledgers, these programs usually amalgamate all three. A typical example would be a program to keep track of the incomings and outgoings of a small trader on a weekly basis. The point of such a system is that it would pull together and summarise the data keyed in from week to week. To do this it has to total up all the cash received and paid out through the week, subtract the one from the other, and present a report of the profit or loss.

The next article in the series will look at the design requirements of such a system.

# CURRENT AFFAIRS

**In the last instalment of the Workshop series we explored some of the phenomena that cause an electrical system to work. Now we look in greater depth at the relationship between the three theoretical components of electricity: potential, current and resistance. To do this we must learn about the way in which they are related, and the very simple piece of arithmetic that predicts that relationship.**

Georg Ohm, a German physicist living in the 19th century, was the first to deduce the relationship between voltage, current and resistance, and to realise that any one of the three fundamental values can be deduced from the other two. Voltage, he found, was equal to the resistance multiplied by the current. Simple arithmetic allowed him to transpose the equation, so that resistance could be determined by dividing the voltage by the current, and current calculated by dividing the voltage by the resistance.

Using the conventional single-letter symbols V for voltage, I for current and R for resistance, the equations themselves are very simple:

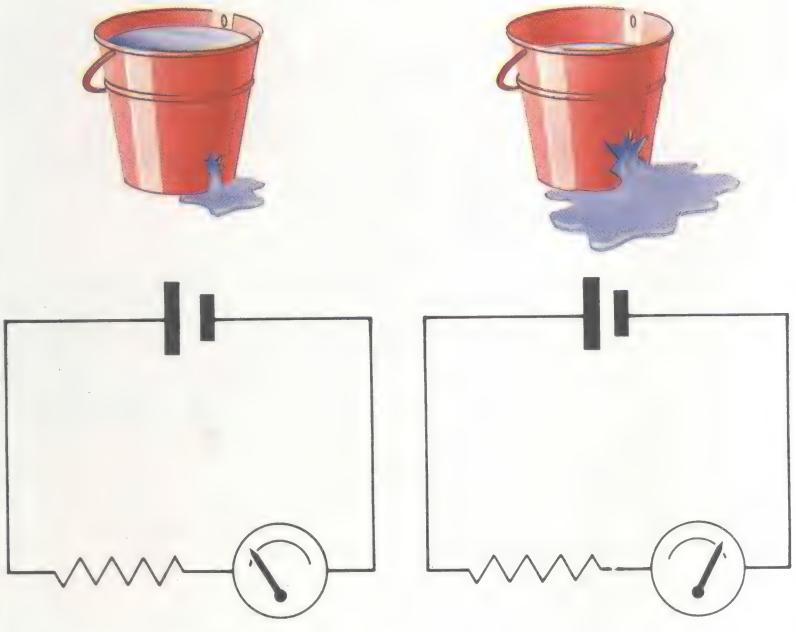
$V = R \times I$
$R = V/I$
$I = V/R$

An extension of Ohm's Law is the Power Law:

$W=V\times I$

which relates W, the power consumed in a circuit or device, to the voltage across it, and the current running through it. If voltage in this formula is expressed in volts, and the current in amps, then W, the power, is measured in watts.

This equation is very useful around the home — it allows us to determine the value of the fuse we should fit to a domestic appliance. Let us take the case of a three kilowatt fan heater running off 240 volts. Dividing 3,000 (the power, in watts, consumed in the device) by 240 (the mains voltage) tells us that the amount of current consumed will be 12.5 amperes. Thus, any fuse less than 12.5 amps value will blow. As the maximum rating for any one household power socket is 13 amps, we can also deduce that any other device running off an adaptor on the same socket as a three kilowatt fan heater must not consume more than 120 watts, or we are endangering the entire circuit. In the next issue we will conduct some experiments to prove Ohm's Law in practice.

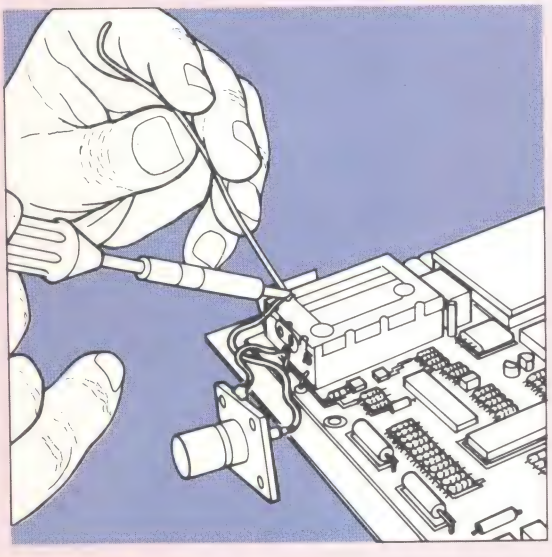


## Resistance And Current

A component in an electrical circuit offers resistance to the flow of electricity, just as the pipes and tanks in a central heating system resist the flow of water: the greater the resistance offered, the less the flow. If a bucket has a small hole, little water will escape because the edges of the hole resist the flow; if the hole is larger, this resistance is lessened, so more water flows. Similarly, an electrical circuit carrying a high resistance shows a lower current flow than the same circuit carrying a low resistance. A component's electrical resistance depends largely upon its material — wires are made of copper because it offers little resistance; light bulb filaments are made of highly resistive material so that the work done by the current against the resistance is expressed in heat, and hence in light

## Quality Control

With this simple modification, the Sinclair Spectrum can provide a composite video signal, which enables the use of a high resolution monitor. Open the case and identify the RF modulator — a large silver component at top left. The modulator has two connections. Find the one nearest the back. Solder a short lead from this to the outside of a surface-mounting BNC socket. Solder another lead from the case of the modulator itself to the central connector on the socket. The socket can be mounted on the side of the Spectrum case



**WARNING: Your computer's guarantee may be rendered null and void if you open the case!**

# Component Parts

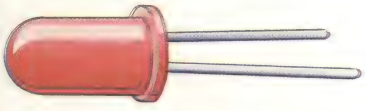
An electronic circuit diagram looks complicated to those who've never seen one. But they are designed to make life easier by providing a way to describe a circuit using a standard set of symbols and techniques. To illustrate this, we've taken a simple circuit for an intercom and described the individual electronic components it uses. On a circuit diagram, each component is represented by a special symbol. Each component on the diagram is also identified by a code, such as 'R1' or 'TR2'. This is a convenient way of referring to the components — in a parts list, for example. The lines connecting the components represent wires. These are drawn straight for neatness — they could be actual wires, or tracks on a printed circuit board

### 1 Switches

There are many varieties of switch, each with its own function. The two main types are latching, where the state of the switch is retained on being pressed, and non-latching, where contact is only made for as long as the pressure is maintained

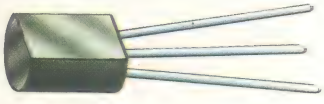
### 2 Light Emitting Diodes

Diodes are the simplest kind of semiconductor. They are the electronic equivalent of a non-return valve, only allowing current to flow in one direction. Some diodes, encapsulated in translucent resin rather than metal, give off a small amount of light, and so find use as indicators



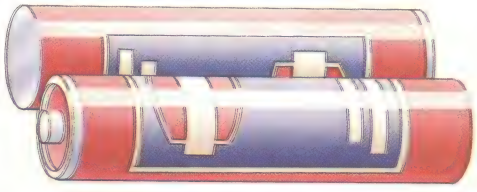
### 3 Transistors

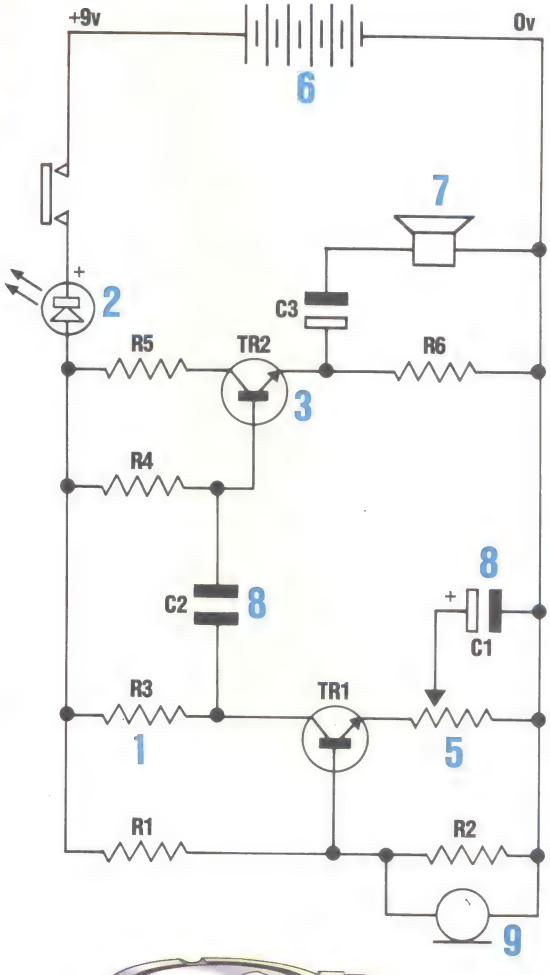
Depending on its specification and the way in which it is used, the transistor can act as a switch or an amplifier. Its invention in 1947 paved the way for subsequent developments in microelectronics

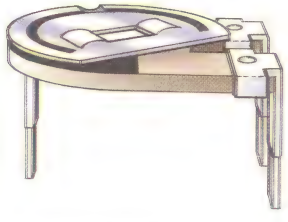


### 4 Resistors

If we introduce less conductive materials into the circuit, we can use them to control the flow of electricity. Resistors come in many different sizes. Their value is expressed by bands of colour round their bodies





### 5 Variable Resistors

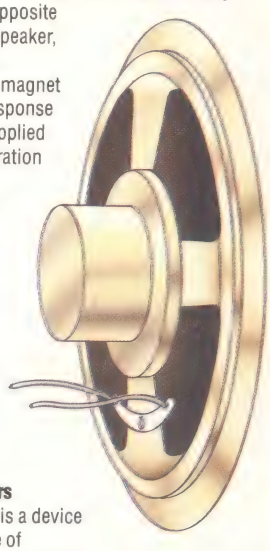
Not all resistors are constant. Variable resistors, sometimes called potentiometers, use a strip of carbon as their conductor. The distance the current has to travel through the carbon before it reaches the take-off terminal determines the resistance of the component

### 6 Batteries

Most small circuits such as this one can be powered with ordinary batteries because they supply a stable direct current

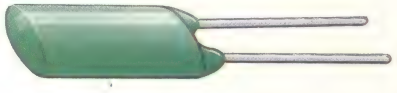
### 7 Speakers

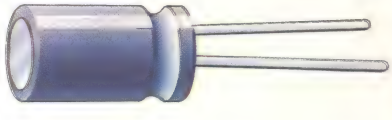
Speakers are close relatives to microphones — they work in the same way but achieve opposite results. In a speaker, a diaphragm attached to a magnet vibrates in response to a current applied to it. This vibration produces sound waves in the air



### 8 Capacitors

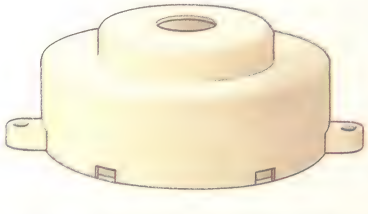
The capacitor is a device that is capable of holding an electric charge. A capacitor is charged up when its two terminals are connected to a power source. Once it is fully charged, nothing further will happen, even when the power is disconnected, until the two terminals of the capacitor become connected, when it will discharge





### 9 Microphones

Microphones work the opposite way to speakers. Sound waves cause the diaphragm to vibrate which, via the magnet, produces different voltages in the circuit



KEVIN JONES

# BYTE BY BYTE

**At this stage in the Machine Code course we follow the full procedure of program development — from defining the initial task through its Assembly language interpretation to the machine code itself — and provide a BASIC program that enables you to enter and observe the results of the code thus created.**

In the previous instalments of the course, we have seen how BASIC program lines are reduced on entry to tokens followed by ASCII data. From this we realised that BASIC, although it is certainly a high-level language, isn't all that high-level: it consists essentially of sequences of instructions, and each instruction consists of a command word (immediately replaced by a token, which is itself only one level above a machine op-code) followed by the data for that command. The fact that the command words and their data (variables, numbers or strings) are closer to natural language, and that the instructions are visibly separated by line numbers or colons, makes a BASIC program look very much more high-level to us than it appears to the BASIC interpreter. It follows, therefore, that machine code should need only a little cosmetic work to make it reasonably comprehensible to our eyes.

The machine code 'cosmetic' that we use is Assembly language, in which alphabetic mnemonics like LDA and ADC stand for the single-byte op-codes that the microprocessor actually understands, and in which alphanumeric symbols such as LABEL1 and TFLAG can be used instead of memory addresses and numeric data. The microprocessor does not understand Assembly language, so before a program can be executed it must be translated into machine code — either by a program called an assembler, or manually by the programmer. The point of Assembly language is that it is machine code in translation. By simply substituting opcodes for mnemonics, and numbers for symbols, it can be turned directly into executable code. But it is much more comprehensible to human eyes than machine code could ever be, and is therefore extremely useful in program development. We shall always write programs in Assembly language, and hardly ever concern ourselves with the machine code equivalent until the very last stages of developing a program. But it's worth doing both at the moment for the sake of interest and for complete clarity, remembering that, in general, Assembly language will do everything we want.

The microprocessor can perform many different operations, but essentially it can only manipulate the contents of memory. It does this by acting directly on computer memory — the RAM and ROM chips that comprise tne computer system — or by working through its own internal memory, which consists of *registers.* These are several bytes of memory physically located inside the microprocessor chip, which have certain special functions, but are otherwise indistinguishable from any other bytes of memory.

## ACCUMULATOR REGISTER

The most important of the microprocessor registers is called the *accumulator.* It is connected directly to the Arithmetic and Logic Unit, and so is used more often than any of the other registers. In order to use it we must be able to put information into it, a process which is called 'Loading the Accumulator'. Using Assembly language, we say that the 6502 does this by performing the LDA operation, and in the Z80 by the operation of LD A. Taking information *from* the accumulator is as essential as loading it, and in 6502 Assembly language this is achieved by the STA (STore the Accumulator contents) operation. The Z80, however, regards both loading and storing as different cases of the same thing — i.e. data transfer. Therefore, taking information from the accumulator register is also done by the LD A operation, but in a different format — as we'll see later in this article.

Suppose, then, that we want to write an Assembly language program that will copy the contents of one byte of memory into the next byte of memory. Let's start by copying byte$09FF into byte$0A00. Immediately, we can express this in Assembly language as:

| 6502 | Z80 |
|---|---|
| LDA $09FF | LD A,($09FF) |
| STA $0A00 | LD ($0A00),A |

Notice that we're copying the contents of byte$09FF into byte$0A00, without knowing what those contents are: it's vital to get this clear from the outset. Byte$09FF may contain any number from $00 to $FF, and all our program does is load that number into the accumulator, then transfer it from the accumulator to byte$0A00. The 6502 version of Assembly language does not make it clear that LDA refers to the *contents* of $09FF, but it does distinguish unequivocally between loading (LDA) and storing (STA). The Z80 version does not make this latter distinction in its opcodes, but its instruction format is always:

OPCODE DESTINATION, (SOURCE)

This version encloses memory addresses in brackets when it means 'the contents of', which reinforces the vital distinction between the address of a byte and what it contains.

The Assembly language program we have given is logically complete, but we shall execute it as a subroutine, so it needs the equivalent of the RETURN command (to refer the program back from the subroutine) to complete it. The opcodes are RTS in 6502, and RET in Z80.

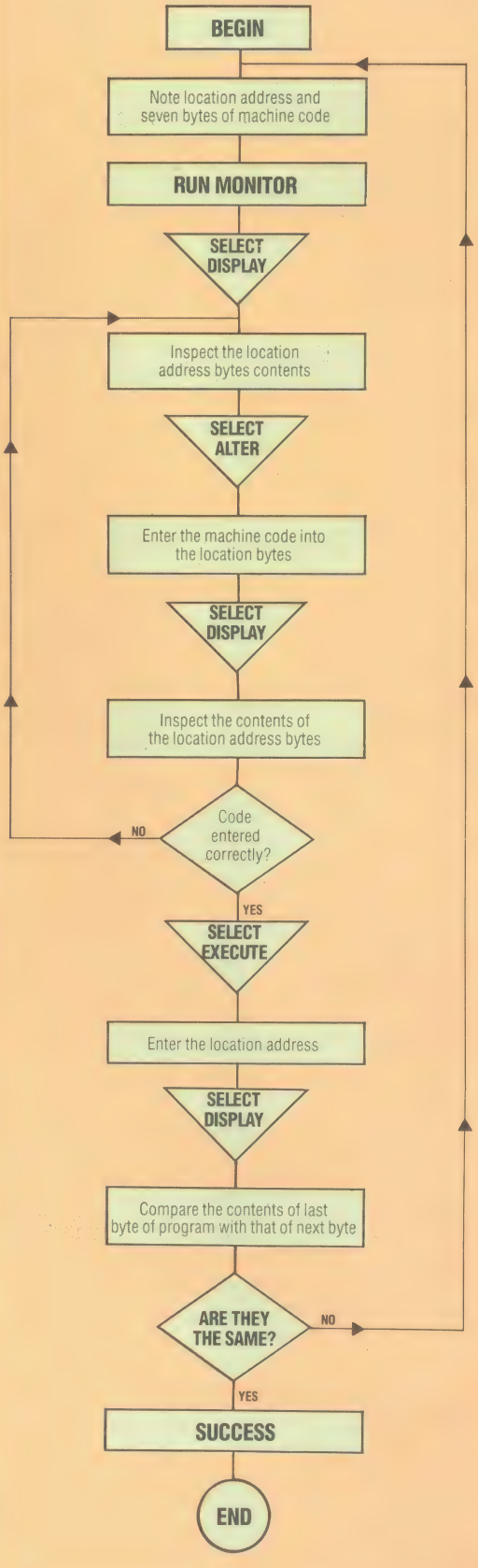
If we are to use this subroutine, we must first translate it into machine code, then store the code somewhere in memory, and cause the microprocessor to execute it. We can use the Monitor program (on page 118) for the latter two tasks; but before that we must first do the translation and decide where the code is to go. This is an unfamiliar decision to BASIC programmers, who never have to think where a BASIC program will be stored — they just type it and RUN it. The storage decisions are made for the programmer by the system designers, and implemented by the operating system.

A machine code program can be stored and executed anywhere in memory, but some places are better than others. The safe places differ from machine to machine, hence the different versions of the program that follow:

| 6502 | | |
|---|---|---|
| **Location Address** | **Machine Code** | **Assembly Language** |
| **COMMODORE 64** | | |
| $0350 | AD 56 03 | LDA $0356 |
| $0353 | 8D 57 03 | STA $0357 |
| $0356 | 60 | RTS |
| **BBC MICRO** | | |
| $0070 | AD 76 00 | LDA $0076 |
| $0073 | 8D 77 00 | STA $0077 |
| $0076 | 60 | RTS |
| **Z80** | | |
| **16K SPECTRUM** | | |
| $7FA0 | 3A A6 7F | LD A,($7FA6) |
| $7FA3 | 32 A7 7F | LD ($7FA7),A |
| $7FA6 | C9 | RET |
| **48K SPECTRUM** | | |
| $FFA0 | 3A A6 FF | LD A,($FFA6) |
| $FFA3 | 32 A7 FF | LD ($FFA7),A |
| $FFA6 | C9 | RET |

Notice that each version of the program copies its own last byte into the byte that comes after it. The Spectrum 48K program, for example, copies the contents of $FFA6 into $FFA7. Notice also that addresses appear in hi-lo form in Assembly language (for our benefit), but lo-hi form in the machine code translation. A special note should be made that in Z80 the mnemonic is LD in both the first and second instructions, but that the opcodes differ: 3A for the data transfer to the accumulator, and 32 for the transfer from the accumulator.

## Using The Monitor Program




The program (on page 118) allows you to **ALTER**, **DISPLAY**, and **EXECUTE** memory. Whenever you select one of these options you will be asked for a hex address. This is the address in memory at which to:
**1)** Start altering memory contents, or
**2)** Start displaying memory contents, or
**3)** Start the microprocessor executing a machine code program.

Entering 'X' instead of a number or address will always return you to command level, and at command level 'Q' will terminate program execution.

When you select **ALTER** mode, and have given the address whose contents you wish to alter, that address will be displayed followed by a query; type the new hex contents and hit **RETURN**. The address of the next byte will be similarly displayed for altering. As long as you wish to alter bytes, keep typing the new contents and hitting **RETURN**. If you enter the letter X instead of an address you will be returned to command level.

**Loading Into Memory**
Look at the program for your machine, note the first location address and the seven bytes of machine code ($0350 and AD,56,03,8D,57, 03,60 on the Commodore 64, for example): you are going to use the Monitor program to load these seven hex numbers into the seven bytes of memory from the last location address onwards.
**1) RUN** the Monitor program, **DISPLAY** the contents of memory from the first location address onwards (from $0350 to $0357 in the Commodore 64, for example).
**2)** Select **ALTER**, enter the location address, and enter the seven bytes that comprise the machine code program.
**3)** Select **DISPLAY** again, and make sure that you have correctly entered the machine code bytes into the location address bytes.
**4)** Select **EXECUTE**, and enter the location address — nothing should appear to happen.
**5)** Select **DISPLAY**, inspect the location addresses, and you should find that the contents of the last byte of the program have been copied into the next byte

## Spectrum Monitor Program

```
48 REM++++++++++++++++++++++
49 REM+                     +
50 REM+  HCAC MONITOR 1     +
51 REM+ ----SPECTRUM------  +
52 REM+ SAVE THIS PROGRAM   +
53 REM+ BEFORE YOU RUN IT   +
54 REM+                     +
55 REM++++++++++++++++++++++
100 GOSUB 1000  :REM *INIT*
200 CLS
300 PRINT " ** HCAC MONITOR 1 CO
MMANDS **"
400 FOR P=1 TO LT:PRINT O$(P):NE
XT P
500 FOR Z=0 TO 1 STEP 0
550 GOSUB 2000  :REM *INPUT*
600 GOSUB (4500+CM*500)
650 NEXT Z
700 STOP
750 REM++++END MAIN PROG++++
1000 REM*****INIT  S/R******
1050 LET LT=4:DIM C$(LT):DIM O$(
LT,24):DIM X$(16)
1100 LET X$="0123456789ABCDEF"
1150 LET C$="ADG@":LET C1=-48:LE
T C2=10-CODE(C$(1))
1200 LET O$(1)="       A - ALTER
  MEMORY"
1220 LET O$(2)="       D - DISPLA
Y MEMORY"
1240 LET O$(3)="       G - EXECUT
E M/CODE"
1260 LET O$(4)="       @ - EXIT
 PROGRAM"
1300 RETURN
2000 REM*****INPUT S/R******
2100 FOR P=0 TO 1 STEP 0
2150 PRINT:PRINT"COMMAND ??"
2190 IF INKEY$<>"" THEN GO TO 21
90
2200 LET A$=INKEY$:IF A$="" THEN
 GO TO 2200
2250 FOR J=1 TO LT
2300 IF A$=C$(J) THEN LET CM=J:L
ET J=LT:LET P=2
2350 NEXT J:NEXT P:IF A$="@" THE
N RETURN
2400 PRINT O$(CM)
2450 FOR P=0 TO 1 STEP 0
2500 INPUT"HEX ADDRESS (X=QUIT)"
;A$
2550 GOSUB 5200 :REM CHK&ADJ
2600 NEXT P:IF A$="X" THEN LET C
M=0
2650 RETURN
3000 REM****HEX BYTE S/R****
3010 LET HB=INT(N/16):LET LB=N-H
B*16
3020 LET B$=X$(HB+1)+X$(LB+1)
3030 RETURN
3100 REM*****D-H   S/R******
3110 IF NM<256 THEN LET N=NM:GOS
UB 3000:LET H$="00"+B$:RETURN
3120 LET HI=INT(NM/256):LET LO=N
M-256*HI
3130 LET N=HI:GOSUB 3000:LET H$=
B$
3140 LET N=LO:GOSUB 3000:LET H$=
H$+B$
3150 RETURN
4000 REM*****H-D   S/R******
4050 LET RX=1:LET DN=0:LET HL=LE
N(H$):IF (HL<1) OR (HL>4) THEN L
ET DN=-1:RETURN
4100 FOR H=HL TO 1 STEP -1
4150 LET D$=H$(H)
4200 LET V=CODE(D$)+C1*(D$>="0"
AND D$<="9") + C2*(D$>="A" AND D
$<="F")
4250 IF V>15 THEN LET DN=-1:LET
H=1:NEXT H:RETURN
4300 LET DN=DN+V*RX:LET RX=RX*16
4350 NEXT H:RETURN
4500 REM*****DUMMY S/R*****
4550 RETURN
5000 REM*****ALTER S/R*****
5020 FOR P=0 TO 1 STEP 0
5040 PRINT A$;:INPUT"NEW HEX VAL
UE (X=EXIT) ?";V$
5050 PRINT V$
5060 GOSUB 5340 :REM CHK&OBY
5080 NEXT P:RETURN
5200 REM*CHECK&ADJUST S/R**
5220 IF A$="X" THEN LET P=2:RETU
RN
5240 LET LL=LEN(A$):IF LL>4 THEN
 RETURN
5260 LET H$=A$:GOSUB 4000
5280 IF DN>=0 THEN LET P=2:LET N
M=DN
5300 LET A$=A$+"  ":IF LL<4 THEN
 LET A$="0000"(TO 4-LL)+A$
5320 RETURN
5340 REM**CHECK & OBEY S/R*
5360 IF V$="X" THEN LET P=2:RETU
RN
5380 LET H$=V$:GOSUB 4000
5400 IF (DN<0) OR (DN>255) THEN
RETURN
5420 POKE NM,DN
5440 LET NM=NM+1:IF NM>65535 THE
N LET P=2:RETURN
5460 GOSUB 3100    :REM D-H S/R
5480 LET A$=H$+"   ":RETURN
5500 REM****DISPLAY S/R*******
5520 FOR P=0 TO 1 STEP 0
5540 INPUT"DEC.NO.OF BYTES(X=QUI
T)";N$:IF N$="X" THEN LET P=2:NE
XT P:RETURN
5560 LET BN=VAL(N$):IF (BN>0) AN
D (BN+NM<65536) THEN LET P=2
5580 NEXT P
5600 FOR B=NM TO (NM+BN-1) STEP
4
5620 LET L$="":LET NM=B:GOSUB 31
00
5640 PRINT H$;TAB(6);
5660 FOR C=0 TO 3
5680 LET N=PEEK(B+C):LET K$="."
5700 GOSUB 3000 :REM D-H S/R
5720 PRINT TAB(6+4*C);B$;
5740 IF N=0 THEN LET K$="■"
5760 IF (N>31) AND (N<128) THEN
LET K$=CHR$(N)
5780 LET L$=L$+K$
5800 NEXT C
5820 PRINT TAB(26);L$
5840 NEXT B:RETURN
6000 REM*****EXECUTE S/R****
6050 RANDOMIZE USR(NM):RETURN
6500 REM*****EXIT  S/R******
6550 PRINT TAB(5);"■■■■END OF PR
OGRAM■■■■"
6600 LET Z=2:RETURN
```

## BBC Micro

```
39 REM*************************
40 REM*    HCAC MONITOR 1     *
41 REM* ---------BBC---------- *
42 REM*   CHANGE THE SPECTRUM  *
43 REM*   VERSION AS FOLLOWS:  *
44 REM*                       *
45 REM*  REPLACE CODE( BY ASC( *
47 REM*                       *
50 REM*  ADD,CHANGE,OR DELETE  *
51 REM*        AS DIRECTED:   *
52 REM*                       *
53 REM*************************
60 *TV 255
70 MODE 7
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="A":C$(2)="D":C$(3)="G":C$(3
)="@":C1=48:C2=ASC(C$(1))-10
2190 ----------DELETE----------
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 ----------DELETE----------
4550 ----------DELETE----------
5050 ----------DELETE----------
5300 A$=A$+"  ":IF LL<4 THEN A$=LEFT$("0
000",4-LL)+A$
5420 ?(NM)=DN
5680 N=?(B+C):K$="."
5740 IF N=13 THEN K$=CHR$(255)
6050 CALL NM:RETURN
6600 Z=1:RETURN
```

## Commodore 64

```
49 REM*************************
50 REM*    HCAC MONITOR 1     *
51 REM* ---------CBM---------- *
52 REM*   CHANGE THE SPECTRUM  *
53 REM*   VERSION AS FOLLOWS:  *
54 REM*                       *
55 REM*  REPLACE ALL INSTANCES *
56 REM*  OF:"LET P=2" BY "P=1" *
57 REM*                       *
58 REM*  REPLACE CODE( BY ASC( *
59 REM*                       *
60 REM*  AND CHANGE OR DELETE  *
61 REM*        AS DIRECTED:   *
62 REM*                       *
63 REM*************************
200 PRINT CHR$(147);CHR$(142)
600 ON CM GOSUB 5000,5500,6000,6500
1050 LT=4:DIM C$(LT),O$(LT)
1150 C$(1)="A":C$(2)="D":C$(3)="G":C$(3)
="@":C1=48:C2=ASC(C$(1))-10
2190 ----------DELETE----------
3020 B$=MID$(X$,HB+1,1)+MID$(X$,LB+1,1)
4150 D$=MID$(H$,H,1)
4500 ----------DELETE----------
4550 ----------DELETE----------
5050 ----------DELETE----------
5300 A$=A$+"  ":IF LL<4 THEN A$=LEFT$("0
000",4-LL)+A$
5740 IF N=0 THEN K$=CHR$(122)
6050 SYS(NM):RETURN
6600 Z=1:RETURN
```

This program will enable you to display the contents of memory, alter the contents of memory and execute a stored machine code program

# DOMESTICATED PET

**Jack Tramiel, Commodore's outgoing chief, has handed over the reins of power to Irving Gould. Tramiel leaves the company in good shape: in the UK alone it claims to have sold nearly half a million Vic-20s and 64s in the second half of last year. What makes its worldwide sales figures the envy of computer makers everywhere?**

The secret of Commodore's success, in the UK as elsewhere, is a well-organised international manufacturing operation, represented in Britain by a modern factory, built with British government grant aid, in the former steel town of Corby. This factory is in the process of being expanded. At present it employs about 250 people and production averages 5,000 computers a day. Like all Commodore plants, it benefits from an assured supply of semiconductor components from the parent company's own factories. Commodore is a huge corporation and because of its volume production it can drive a hard bargain with outside suppliers: in some cases it pays only half what its competitors pay for vital chips.

Commodore's strong position owes much to the success of the CBM PET — Personal Electronic Transactor. Its design was essentially that of Chuck Peddle, who made three important decisions. The first was to base the machine on the 6502 processor; the second was to equip it with Microsoft BASIC; the third was to provide a full screen editor, which made the machine much easier to use than the single-boards that microelectronics enthusiasts had been playing with since the advent of the microprocessor in the mid-1970s. To this day, the supposedly 'next generation' IBM PC lacks a full screen editor as standard.

Commodore was in a strong position to build such a machine — it already owned MOS Technology, which had the rights to produce the 6502 microprocessor. This was invaluable: both the PET's competitors, the Apple II and the Tandy TRS-80, used the 6502 CPU, so Commodore was well-placed to monitor their production. The birth of the PET was nevertheless problematic. Jack Tramiel insisted that the PET's memory components also derive from MOS Technology, against Peddle's wishes. This led to a highly publicised row between the two men and Peddle's abrupt departure from the company.

The PET is now a venerable piece of machinery. Originally housed in a pressed steel box (in the way office furniture used to be built) it has now been re-skinned in plastic to bring its appearance up to date. In its latest guise, with the potential to handle a 22 Megabyte disk store, it is called the 8000 series. Despite its age, it is still selling remarkably well to more conservative customers, who feel comfortable with it and see no reason to change their software for the new generation of office computers based on the 8088 CPU. Its suppliers even claim that — to their own amazement — it is still competitive with the IBM PC and ACT's Apricot.

This may be the result of a historical accident. Commodore, with Apple and Tandy, were the first companies to introduce an affordable and easily used personal computer. But once it had got its customers, Commodore hung on to them. From conservatism and a desire to contain development costs, it never bothered to overhaul the specification of its beginner's machine. Most of the original PET software will still run on today's machines, and the Microsoft BASIC Version 2.0 is still much as it was when Peddle adopted it.

Unfortunately for the more advanced hobbyist, Microsoft 2.0 was developed at a time when graphics and sound were luxuries on a cheap computer. Though Commodore's recent hobby computers are well-equipped with these features, the BASIC lacks the necessary commands, and requires extensive and laborious use of POKEs to address specific memory locations. This, however, is not a problem with pre-written cartridge software, of which Commodore offers a wide range.

It is at least partly due to its steady supply of components that Commodore managed to ride the storm that sank its competitors in the games market. Texas and Mattel withdrew altogether from home computers, and Atari has hardly looked shipshape in the last couple of years. Commodore, with its assured supply of cheap parts, was able to drive the prices of computer and cartridge software down below the price at which other manufacturers could compete, still making a profit. At one point, Commodore's cartridges were a third the price of those of its competitors. Modern, automated plant and access to cheap parts were the results of two decades' experience in manufacture. The factory gate price of a Commodore 64 is reportedly as little as £38.

Commodore has not worked itself into this dominating position by sheer luck. It has had a rough ride and has stood on the verge of bankruptcy at least twice. Jack Tramiel, who arrived in the United States after the Second World War as a teenage Polish refugee from Auschwitz, formed CBM — Commodore

COURTESY OF MICROSCOPE

**Jack Tramiel**
The driving force behind Commodore has been its president Jack Tramiel. His shrewd marketing ability will be missed now he has left the company full-time

COURTESY OF VNU

**Chuck Peddle**
Chuck Peddle is the man behind the PET design and the 6502 chip inside it. He went on to form his own company to produce the Sirius 16-bit business machine

Business Machines — in 1955. Its base was Toronto, Canada, where the fledgling company started its manufacturing operation in a modest way by assembling typewriters under licence from Czechoslovakia. He chose the name, it is said, because of its similarity to IBM.

In 1975, after two decades of trading in office products, the company was brought to its knees by the ferocious calculator wars, which the Japanese eventually won. But Tramiel, feared as much as respected for his business methods, was a survivor. He spotted the potential market for a personal computer and in 1976 brought Chuck Peddle into the company and in under a decade saw the value of the company grow 50-fold.

Tramiel has made Commodore into a redoubtable trading and manufacturing operation but, if it has a weakness, it is in new product development. The company's philosophy is 'We sell to the masses, not the classes' and Tramiel's belief has been that the customer will always buy what offers best value for money. The requirements of cheap volume manufacture may militate against incorporation of the most advanced technology. In late 1982 a large proportion of Commodore's small research and development staff left the company in a mass walk-out, and since then it has relied on buying in the fruits of the researches of other companies. It has entered manufacturing agreements with Far East concerns for disk drives, and has held talks with firms such as Sony to buy in expensive 'fifth-generation' technologies such as voice recognition, home robots and sophisticated storage devices. Commodore even approached Oric's innovative designer Paul Johnson to see if he would design a ULA chip for its new series of home computers.

In 1984 Commodore seems more confident than ever and continues to make a virtue of cheapness and simplicity. It has exhibited two new home computers based on the new 7501 processors and known as the 264 and V364. The latter has a speech synthesiser with a built-in vocabulary of 250 words. In line with current trends, software for word processing, spreadsheet calculation and graphics will be available as an option. The V364 sounds as though it will be an entry into the market for home computers that other micro makers will ignore at their peril.

## Commodore Milestones

**1984**
The **SX64** is an updated version of the 64 in a portable case with a colour screen and disk drive

**1977**
The original **Commodore PET** was the first mass-market personal computer. After numerous updates, it is still selling well

**1982**
The elusive **CBM 700** is the much delayed replacement for the 8032 machines that promises to bring Commodore's business machines in line with more modern micros

**1979**
The **Vic-20** was Commodore's first cheap home computer, but despite limitations and strong competition is still very popular

**1981**
The **CBM 8032** added an 80-column ability to the PET range, giving it the ability to run serious business software

**1983**
The **Commodore 64**, having a 40-column screen and a 64K memory, has improved on the restrictions of the Vic

**1980**
The **SuperPET** (or CBM 9000) was an attempt to produce a dramatically changed business version of the PET

IAN McKINNELL, CHRIS STEVENS AND COURTESY OF COMMODORE

# Mentathlete


Home computers. Do they send your brain to sleep – or keep your mind on its toes?

At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.

Provided, of course, it offers a whole battery of genuine mental challenges.

The Spectrum does just that.

Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!

The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.

And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.

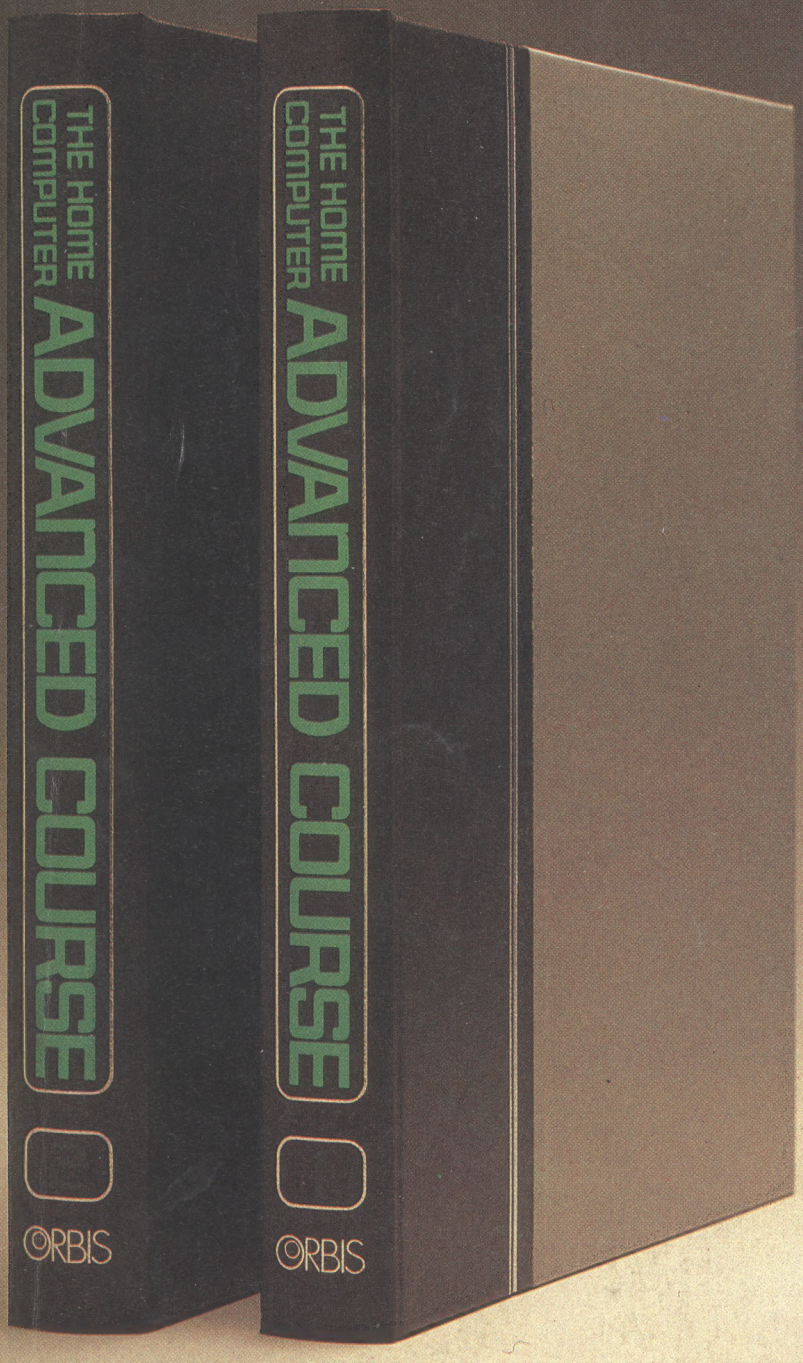
No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.

For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.

sinclair

THE HOME COMPUTER ADVANCED COURSE

# WE HAVE DESIGNED BINDERS SPECIALLY TO KEEP YOUR COPIES OF THE 'ADVANCED COURSE' IN GOOD ORDER.




**A**ll you have to do is complete the reply-paid order form opposite – tick the box and post the card today – **no stamp necessary!**

**B**y choosing a standing order, you will be sent the first volume free along with the second binder for £3.95. The invoice for this amount will be with the binder. We will then send you your binders every twelve weeks – as you need them.

**Important:** This offer is open only whilst stocks last and only one free binder may be sent to each purchaser who places a Standing Order. Please allow 28 days for delivery.

**Overseas readers:** This free binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain binders now. For details please see inside the front cover. Binders may be subject to import duty and/or local tax.

**The Orbis Guarantee:** If you are not entirely satisfied you may return the binder(s) to us within 14 days and cancel your Standing Order. You are then under no obligation to pay and no further binders will be sent except upon request.

## PLACE A STANDING ORDER TODAY.